```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
cd /content/drive/MyDrive/CSE475/project1
```

```
/content/drive/MyDrive/CSE475/project1
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pylab as pl
import scipy.optimize as opt
from sklearn import preprocessing
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import accuracy_score
```

```
dataset = pd.read_csv('covid_dataset.csv')
dataset
```

```
                Day  Lab Test  Confirmed case  Death Case
```

```
x= dataset [['Confirmed case']]
y= dataset[['Death Case']]
x_train , x_test ,y_train,y_test = train_test_split (x,y,test_size = 0.3 , random_state = 42
```

```
      2   2020-04-06       466              35          3
```

```
dataset.shape
```

```
     (626, 4)
```

```
      ...         ...        ...             ...         ...
```

```
X = np.asarray(dataset[['Confirmed case']])
X[0:5]
```

```
     array([[ 9],
            [18],
            [35],
            [41],
            [54]])
```

```
Y= np.asarray(dataset[['Death Case']])
Y[0:5]
```

```
     array([[2],
            [1],
            [3],
            [5],
            [3]])
```

```
print ('Train set:', x_train.shape,  y_train.shape)
print ('Test set:', x_test.shape,  y_test.shape)
```

```
     Train set: (438, 2) (438, 1)
     Test set: (188, 2) (188, 1)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=0.01, solver='liblinear').fit(x_train,y_train)
LR
```

```
     /usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConversionWa
       y = column_or_1d(y, warn=True)
     LogisticRegression(C=0.01, solver='liblinear')
```

```
yhat = LR.predict(x_test)
yhat
```

```
     array([ 35, 237,  21,  35, 237,  35,  35,  35,  21,  35,  21,  35,  35,
        35,   3,  35,  21,  35,  21,  21,  35,  35,  35,   6,  35,  35,
        21,   6,   6,  21,  16,  21,  21, 237,  35, 241,  16,   6,  21,
         6,  21,  35,  35,   6,   7,  21,  21,  21,  35,  35,   7,  21,
        21,  21,   3,   7,  35,   6,  21,  21,   6, 237,  21,   6,   6,
        16,  35,  21,  21,  35,  21,  21,  35,  21,  16,  21,   6,  35,
       237,  21,  35,  35,  21,  35,  21,   6,  35,   7,  21,  35,  21,
         7,  21,  35,  35,  21,  16,  21,  21,  21,  21,  35,  21,  21,
        35,  35,  21,  35,  35,  35,  35,  35,  35,  35,   6,  35,   6,
         7,   6,  21,  35,  21,  35,   3,  21,  21, 237,  21,  35,  35,
       237,  21,  35,   6,  21,  16,  21,   3,  21,   3,  21,   6,  35,
        35,  16,  35,  21,  16,  35,   7,  21,  35,  21,  35,  35,  21,
        21,   7,  21, 237,  16,  21,  35,  21,  35,  21,  21,  21,  35,
        21,   3,  21,  35,  16,   7,   7,  21,  16,  35,  35,  16,  35,
        35,  35,   7,  35,  21,  35])
```

```
yhat_prob = LR.predict_proba(x_test)
yhat_prob
```

```
array([[9.71e-072, 9.58e-027, 1.91e-016, ..., 4.42e-003, 1.47e-003,
        1.83e-003],
       [1.07e-260, 2.12e-095, 1.43e-057, ..., 2.18e-002, 1.42e-002,
        4.21e-003],
       [5.91e-030, 1.49e-011, 2.47e-007, ..., 9.70e-005, 1.11e-005,
        3.15e-005],
       ...,
       [6.99e-053, 5.85e-020, 2.02e-012, ..., 2.65e-003, 8.01e-004,
        1.17e-003],
       [4.15e-007, 5.34e-004, 2.63e-003, ..., 5.96e-003, 4.29e-003,
        5.07e-003],
       [1.79e-073, 7.69e-027, 3.68e-016, ..., 4.58e-005, 2.77e-006,
        8.74e-006]])
```

```
LR.score(x_test,y_test)
```

```
0.031914893617021274
```

```
from sklearn.metrics import classification_report, confusion_matrix
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print(confusion_matrix(y_test, yhat, labels=[1,0]))
```

```
[[0 0]
 [0 0]]
```

```
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, yhat, labels=[1,0])
np.set_printoptions(precision=4)


# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=['Death Case=1','Death Case=0'],normalize= False,
     Confusion matrix, without normalization
     [[0 0]
```

```
print (classification_report(y_test, yhat))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00         3
           2       0.00      0.00      0.00         5
           3       0.00      0.00      0.00         5
           4       0.00      0.00      0.00         3
           5       0.00      0.00      0.00         5
           6       0.12      0.40      0.18         5
           7       0.00      0.00      0.00        11
           8       0.00      0.00      0.00         3
           9       0.00      0.00      0.00         2
          10       0.00      0.00      0.00         2
          12       0.00      0.00      0.00         1
          13       0.00      0.00      0.00         3
          14       0.00      0.00      0.00         2
          15       0.00      0.00      0.00         2
          16       0.08      0.25      0.12         4
          17       0.00      0.00      0.00         6
          18       0.00      0.00      0.00         4
          19       0.00      0.00      0.00         3
          20       0.00      0.00      0.00         3
          21       0.03      0.67      0.06         3
          22       0.00      0.00      0.00         1
          23       0.00      0.00      0.00         4
          24       0.00      0.00      0.00         2
          25       0.00      0.00      0.00         5
          26       0.00      0.00      0.00         3
          27       0.00      0.00      0.00         4
          28       0.00      0.00      0.00         4
          29       0.00      0.00      0.00         1
          30       0.00      0.00      0.00         3
          31       0.00      0.00      0.00         2
          32       0.00      0.00      0.00         5
          33       0.00      0.00      0.00         2
          34       0.00      0.00      0.00         2
          35       0.01      0.50      0.03         2
          36       0.00      0.00      0.00         4
          37       0.00      0.00      0.00         8
          38       0.00      0.00      0.00         4
          39       0.00      0.00      0.00         4
          40       0.00      0.00      0.00         3
          41       0.00      0.00      0.00         2
          42       0.00      0.00      0.00         3
          44       0.00      0.00      0.00         2
          45       0.00      0.00      0.00         5
          46       0.00      0.00      0.00         1
          50       0.00      0.00      0.00         2
          51       0.00      0.00      0.00         1
          53       0.00      0.00      0.00         1
          55       0.00      0.00      0.00         1
          58       0.00      0.00      0.00         1
          60       0.00      0.00      0.00         1
          63       0.00      0.00      0.00         1
          66       0.00      0.00      0.00         1
          69       0.00      0.00      0.00         2
          70       0.00      0.00      0.00         1
```